УДК 004.89

АНАЛІЗ ТЕХНІЧНИХ ТА ЛІНГВІСТИЧНИХ БАР'ЄРІВ ПРИ ПОБУДОВІ БАГАТОМОВНИХ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ

Китайченкова К.С.[1]; Трофименко О.Г. к.т.н., доц.[2]
[1,2] Національний університет «Одеська юридична академія»
[1,2] Україна, Одеса
[1]ekaterinakitajcenkova@gmail.com; [2]ORCID: 0000-0001-7626-0886

Анотація. *Досліджено вплив конвеєрного та наскрізного архітектурних підходів на ефективність багатомовних інтелектуальних агентів. Показано, що модульна інтеграція машинного перекладу створює «семантичне вузьке місце», яке ізолює логіку агента від оригінального запиту користувача, що призводить до втрати контексту та лінгвістичних нюансів.*
Ключові слова*: штучний інтелект; обробка природної мови; машинний переклад; багатомовні інтелектуальні агенти; архітектура чат-ботів; технічні бар'єри; лінгвістичні бар'єри; семантичне вузьке місце.*

Вступна частина. Стрімка еволюція розмовного штучного інтелекту (ШІ) знаменує перехід від простих скриптованих чат-ботів до складних, автономних інтелектуальних агентів. Цей прогрес підсилений великими мовними моделями (Large Language Models, LLM) – класом моделей ШІ, навчених на величезних масивах даних для розуміння та генерації людиноподібного тексту, – сформував у користувачів очікування безшовної, інтуїтивної та людиноподібної взаємодії. За умов глобалізації корисність та універсальність таких агентів визначається їхньою здатністю долати мовні бар'єри. Тому розробка багатомовних систем перетворилася з нішевої функціональності на фундаментальну вимогу для забезпечення глобального охоплення та доступності технологій.

Мета роботи – проаналізувати технічні та лінгвістичні бар'єри, які виникають при побудові багатомовних інтелектуальних агентів.

Основна частина. На перший погляд, шлях до створення багатомовного агента видається очевидним: інтеграція наявного сервісу машинного перекладу (МП) у вигляді окремого модуля в архітектуру системи. Такий підхід, відомий як конвеєрний або модульний, є розповсюдженим через його уявну простоту та можливість використання готових API-рішень [1]. Але проблемою конвеєрного підходу є створення так званого «семантичного вузького місця» (semantic bottleneck) [2]. Логічне ядро агента, яке є його «мозком», повністю ізольоване від оригінального, неспотвореного запиту користувача. Воно оперує не з першоджерелом, а з його вторинною, «санітизованою» інтерпретацією, наданою системою МП. Цей проміжний етап діє як фільтр, який відсікає критично важливі шари інформації: культурні нюанси, ідіоматичні вирази, сарказм, лексичну неоднозначність та специфічний стилістичний тон. Як наслідок, NLU-модуль (Natural Language Understanding) – компонент, відповідальний за розуміння природної мови шляхом розпізнавання наміру користувача та вилучення ключових сутностей із запиту, - змушений ухвалювати рішення на основі збіднених, а іноді й хибних даних, що призводить до генерації відповідей, які сприймаються як роботизовані, нерелевантні або логічно некоректні [3].

Цей конфлікт між простотою реалізації та якістю взаємодії формує ключовий дослідницький виклик. Він охоплює як технічні бар'єри (підвищену затримку відповіді (latency) через множинні мережеві запити та ефект каскадного поширення помилок), так і

складні лінгвістичні перешкоди, пов’язані зі збереженням контексту в багатоетапних діалогах та коректною обробкою небуквальних мовних конструкцій.

Фундаментальні проєктні рішення, прийняті на етапі розробки, визначають не лише можливості та масштабованість агента, а і його вразливі місця. Для побудови багатомовних систем є два фундаментальні архітектурні підходи: конвеєрний та наскрізний [4].

Конвеєрна архітектура (Pipeline Architecture) реалізує систему як послідовність ізольованих етапів обробки [5]. Її «семантичне вузьке місце» проілюстровано на прикладі розробленого авторами Telegram-бота «Fluentia». У цьому проєкті зовнішній Google Gemini API використовується як окремий модуль для перекладу. Щоб отримати якісний результат (переклад з контекстним прикладом, адаптованим до рівня користувача), необхідно створювати надзвичайно деталізовані запити (промпти) до API (рис. 1). Це є компенсаторним механізмом, який намагається подолати вразливість самої архітектури, змушуючи розробника вручну інструктувати зовнішній сервіс, щоб той зберіг контекст і повернув дані у строгому форматі.

```
async def translate_with_example_gemini(text_to_translate: str,
    user_level : str | None = None)->str:
// ... ініціалізація API ...

// Формування детального промпту для компенсації втрати контексту
prompt = (
    f"Проаналізуй текст: '{text_to_translate}'.\n"
    f"Завдання: Надай переклад та, якщо ТИП ТЕКСТУ = ОДИНИЧНЕ СЛОВО або ІДІОМА,
то також приклад використання.\n"
    f"1. Визнач мову оригіналу (МОВ ОРГ: українська або англійська).\n"
// ... ~15 рядків детальних інструкцій щодо мови, формату та регістру...
    f"8. **СТРОГО ВИВЕДИ РЕЗУЛЬТАТ В ОДНОМУ З ДВОХ ФОРМАТІВ НИЖЧЕ...**\n"
    )
// ... відправка запиту та обробка відповіді ...
```

Рисунок 1 – Приклад формування детального промпту для Gemini API

Наскрізна архітектура (End-to-End Architecture), на противагу модульній, використовує єдину, попередньо навчену багатомовну модель (наприклад, mBERT, XLM-R), що обробляє декілька мов безпосередньо [6]. Завдяки технології крос-лінгвальних вкладень (cross-lingual embeddings), така модель створює єдиний семантичний простір для різних мов, що дозволяє їй «розуміти» запити без проміжного етапу перекладання. Такий підхід усуває «семантичне вузьке місце», але вимагає значних обчислювальних ресурсів. Вибір наскрізної архітектури стає джерелом комплексу взаємопов’язаних технічних та лінгвістичних бар’єрів. На технічному рівні першочерговою проблемою є кумулятивна затримка (latency) [7], оскільки кожен хід діалогу вимагає щонайменше двох мережевих запитів до API, що порушує природну динаміку розмови. Крім того, виникає ефект каскадного поширення помилок (error cascade) [8], за якого навіть незначна неточність на етапі перекладу провокує хибну інтерпретацію наміру на етапі NLU, що призводить до нерелевантних відповідей. У свою чергу, ці технічні вади безпосередньо спричиняють глибокі лінгвістичні проблеми. Втрачаючи оригінальний контекст під час перекладу, система втрачає здатність розв’язувати лексичну та синтаксичну неоднозначність. Аналогічно, вона виявляється нездатною коректно обробляти ідіоми, метафори та культурні нюанси, що призводить до генерації буквальних, але семантично безглуздих і культурно недоречних відповідей.

Проведений аналіз демонструє, що створення ефективних багатомовних інтелектуальних агентів є не стільки задачею інтеграції зовнішніх сервісів, скільки фундаментальною архітектурною проблемою. Ключовим висновком є те, що поширена конвеєрна архітектура, попри свою інженерну привабливість, неминуче створює «семантичне вузьке місце», яке ізолює NLU-ядро агента від оригінального, багатого на нюанси запиту користувача. Потреба у формуванні складних, деталізованих промптів до Google Gemini API для збереження контексту не є ознакою потужності, а скоріше – вимушеним компенсаторним механізмом, який лише частково нівелює вразливості самої архітектури.

Висновки. Зазначена фундаментальна архітектурна вада безпосередньо генерує каскад взаємопов'язаних проблем: від технічних, як-от кумулятивна затримка (latency) та поширення помилок (error cascade), до глибоких лінгвістичних, які проявляються у нездатності системи коректно обробляти лексичну неоднозначність, ідіоми та культурні нюанси. Шлях до створення по-справжньому глобальних та ефективних інтелектуальних агентів лежить через відмову від модульних, послідовних систем на користь цілісних, наскрізних (end-to-end) архітектур, здатних обробляти мову в її оригінальному вигляді. Це можливо лише за умови, що багатомовність розглядається не як додаткова функція, а як базовий принцип, закладений в основу архітектури з самого початку розробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Vasan, A. (2025). Why multilingual agentic AI is key to global customer experience. https://www.parloa.com/knowledge-hub/multilingual-agentic-ai/

[2] Зосим, М. (2023). Вузьке місце у виробництві Bottleneck (production). https://www.maxzosim.com/vuzkie-mistsie-u-virobnitstvi/

[3] Нельсон, Д. (2020). Що таке NLU (Розуміння природної мови)? (2020). https://www.unite.ai/uk/what-is-natural-language-understanding/

[4] Konyk, M., Vysotska, V., Goloshchuk, S., Holoshchuk, R., Chyrun, S., & Budz, I. (2023). Technology of Ukrainian-English Machine Translation Based on Recursive Neural Network as LSTM. *International Conference on Computational Linguistics and Intelligent Systems*. https://www.semanticscholar.org/paper/Technology-of-Ukrainian-English-Machine-Translation-Konyk-Vysotska/69853d672d85b2fb054cee29a15728cf0d2e0c31

[5] Pipeline Architecture. https://www.cs.sjsu.edu/~pearce/modules/patterns/distArch/pipeline.htm

[6] Gupta, V. (2025). End-to-End Infrastructure Architecture: A Technical Overview. *Medium.* https://vishal-vishal-gupta48.medium.com/end-to-end-infrastructure-architecture-a-technical-overview-c34534cc1ccf

[7] Що таке Latency: Як затримка впливає на роботу інтернету та додатків. https://www.run-it.com.ua/shcho-take-latency-iak-zatrymka-vplyvaie-na-robotu-internetu-ta-dodatkiv/

[8] Zellinger, M. & Thomson, M. (2025). Rational Tuning of LLM Cascades via Probabilistic Modeling. *arXiv*. https://doi.org/10.48550/arXiv.2501.09345

Kytaichenkova K.S., Trofymenko O.G.

Analysis of technical and linguistic barriers to building multilingual intelligent agents

Abstract. *The impact of pipelined and end-to-end architectural approaches on the efficiency of multilingual intelligent agents is investigated. It is shown that the modular integration of machine translation creates a "semantic bottleneck" that isolates the agent logic from the original user request, leading to a loss of context and linguistic nuances.*

Keywords: *artificial intelligence; natural language processing; machine translation; multilingual intelligent agents; chatbot architecture; technical barriers; linguistic barriers; semantic bottleneck.*